PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04L 29/06, G06F 9/46

A1

(11) International Publication Number: WO 98/26553

(43) International Publication Date: 18 June 1998 (18.06.98)

(21) International Application Number: PCT/US97/22117

(22) International Filing Date: 6 December 1997 (06.12.97)

(30) Priority Data:
08/763,289 9 December 1996 (09.12.96) US

(71) Applicant: SUN MICROSYSTEMS, INC. [US/US]; 901 San Antonio Road, M/S Pal1-521, Palo Alto, CA 94303 (US).

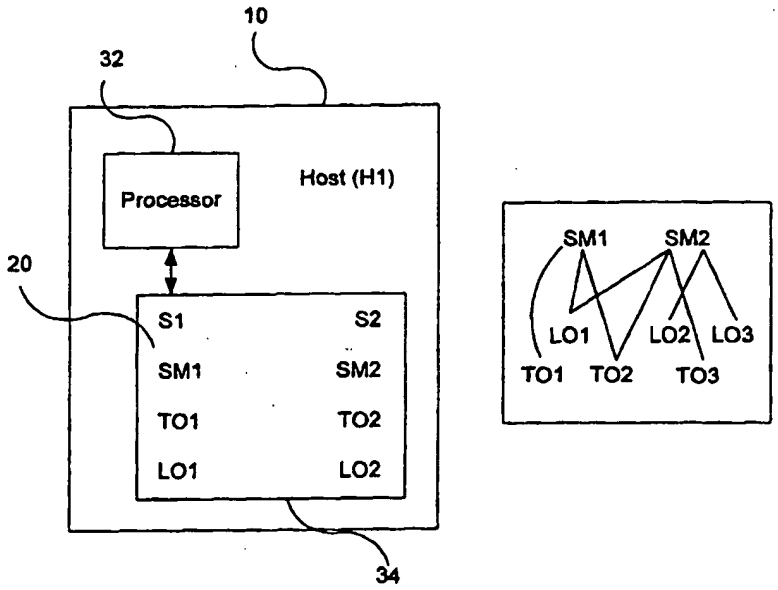
(72) Inventors: LIM, Swee, B.; 11691 Timber Spring Court, Cupertino, CA 95014 (US). SINGHAI, Ashish; Room 3234, 1203 W. Springfield Avenue, Urbana, IL 61801 (US). RADIA, Sanjay, R.; 883 Boar Circle, Fremont, CA 94539 (US).

(74) Agents: MAJERUS, Laura, A. et al.; Graham & James LLP, 600 Hansen Way, Palo Alto, CA 94304 (US).

(81) Designated States: JP, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: LOAD BALANCING AND FAILOVER OF NETWORK SERVICES



(57) Abstract

A system and method for many-to-many failover and load balancing establish a plurality of service groups for providing desired computing services. Each service group comprises a plurality of hosts, and each host within the service group is available to perform the computing services for which that group as a whole is responsible. Each host may belong to a plurality of service groups. Each operational host within a service group transmits periodic messages to each other host within the service group advising of the status of the transmitting host. A leader host evaluates the periodic messages and, where appropriate, dynamically reassigns responsibility for particular computing services to a host within the group. The reassignment can be due either to failover or load balancing.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Larvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

LOAD BALANCING AND FAILOVER OF NETWORK SERVICES

FIELD OF THE INVENTION

This application relates to systems and methods for management of services within a network of computer systems, and particularly relates to load balancing and host failover within such a network.

BACKGROUND OF THE INVENTION

In providing reliable computer services, and especially for Internet applications such as HTTP, FTP, News and email, a fundamental requirement is a platform for providing those services which is both scaleable and reliable.

There are two kinds of scaleability: vertical and horizontal. Vertical scaleability is best characterized by the IBM paradigm of the 1970s and 1980,s, in which a company's growing need for computer services meant that a less powerful computer was replaced in its entirety by a more powerful computer. This paradigm has been substantially discredited for a number of reasons, including the fact that it is limited to whatever is the most powerful hardware available and because of the expense of such a single machine becomes prohibitive. Such machines are inherently not optimal in price/performance. The lack of reliability of a single machine is also a key limitation in vertical scaleability.

Horizontal scaleability, in contrast, adds more computer systems as load increases. Each of these computers is typically less powerful than the vertically scaleable solution, but the combined power of multiple such systems frequently exceeds that of the vertical solution for many applications. The horizontal solution also permits the user to maximize the cost benefit of prior investments (i.e., prior purchases of still-compatible computer systems). Horizontal scaleability can therefore be seen to offer a number of-key advantages over vertical scaleability.

Reliability is also best served by a horizontally scaleable solution. Reliability is measured by the availability of computer services when needed; since no single computer has ever proved to have 100% up-time, reliability

requires more than one computer capable of providing the needed computer services.

As the need for continuously available computer services has grown, the need for increased scaleability and {reliability has also grown. One of the key issues has been to ensure that a service provided- by a first computer, normally termed a host, can be provided by another computer, or a backup, in the event the host becomes unavailable. This transfer of services is termed failover, and in current systems is typically handled by software.

Two failover schemes are well-known in the prior art. One-to-one failover designates a host system as primary and a backup system as secondary; in the most classic implementation of this approach, the secondary system is idle -- that is, it provides no services -- until the host fails. When the host becomes unavailable, the secondary system provides the services normally provided by the host. Symmetric one-to-one failover is a similar technique, wherein each of the "host" and "backup" systems provide distinct but useful sets of services when both are available, and each is capable of providing the services normally provided by the other. Thus, each system is both a primary and a secondary, but only the one machine can serve as a backup to the other.

The second failover scheme known in the prior art is many-to-one failover. In this approach there are many primary systems but only a single secondary, with each of the primaries providing a distinct set of services. The secondary or backup system is capable of performing any of the services provided by any or all of the primaries, but normally sits idle until a primary fails.

Each of these schemes is limited in that the networks are reliable only as long as only one system fails; network services become unavailable if more than one system becomes unavailable. In addition, these systems do not allow for good failover scaleability because the secondary system typically must be identified at initial configuration and cannot thereafter be changed. Along this same line, prior art systems do not allow failed hosts to be permanently Reinstalled, nor do they allow new hosts to be added and configured without

reconfiguring existing hosts. An additional limitation of such prior art techniques is the inability to perform load balancing.

There has therefore been a need for an improved failover system in which computing services continue to be available over the network even when more than one host or primary system has failed, and in which hosts may be added or removed without reconfiguring the remainder of the systems forming the network.

SUMMARY OF THE INVENTION

The present invention overcomes virtually all of the limitations of the prior art by providing a system and method by which computing services may be provided by any of a plurality of systems within a group, such that the computing services provided by that group will continue to be available as long as at least one system is available within the group. The system and method involves assignment of IP addresses to the hosts and services and management of those IP addresses to achieve the functionality described below.

To provide many-to-many failover, a plurality of hosts are organized into a Service group wherein each host is capable of providing the service rendered by the service group. Each host may also be available to provide other services, and thus may be members of other service groups. At any given time, a service is provided by all of the active members of the service group. Thus, it is important to maintain communications within the group and to coordinate the cooperation of the various hosts to ensure that the service provided by that group remains available, even if one or more hosts providing the service become unavailable from time to time, such as through failure or load balancing.

To maintain communications among the hosts within the service group, each host periodically sends a "Control message" to all other hosts within the group. The control message may also be thought of as an "info message" or, for some purposes, a .heartbeat. The control message basically keeps each other host appraised of the status of the sending host as well as data about the perceived status of the other hosts within the group.

To coordinate the cooperation of the various hosts within the service group, a Leader host is established and assigns to the various hosts responsibility

for particular service addresses. The leader identifies resources, such as service addresses (which may, for example, be IP addresses) which are not being attended and causes such resources to be acquired by designated hosts. The leader also dynamically reassigns resources to hosts which have been newly added or restored to the service group, as well as causing the release of resources handled by failed hosts.

Load balancing can also be accomplished with the present invention, either through statistical load-balancing by measuring utilization and using round-robin rotation of entries in a Domain Name Service (DNS) zone, or through DNS zone modification.

Utilization may be determined as a ratio of load (expressed, for example, in instructions per second not idling) divided by current capacity (typically expressed in terms of instructions per second) and for each host is calculated by a service-specific module, since each service supported by a host may have different load and capacity. By comparing the utilization with a preset maximum utilization setting (which may be referred to as a ..high water, mark), load shedding can be initiated if a hosts utilization exceeds that high water mark by shifting new connections to other less-loaded hosts. When a hosts utilization drops below a preset minimum utilization (or low water, mark), that host can be added back to the E)NS zone.

DNS zone modification can be used for load balancing by establishing a configuration parameter that specifies the minimum number of available service addresses that must be present in a DNS zone. By such an approach, highly loaded hosts that exceed the high water mark may continue to be identified as available.

Other features of the system of the present invention include provisions for maintaining hosts as hot spares; preventing a host from acquiring service addresses - i.e., maintaining a host in a "quiesced" state; and failure detection for either a total host failure or a service failure.

In the foregoing manner the management of service addresses in accordance with the present invention can be seen to provide computing services

in a new and novel manner. This and additional advantages will be better understood from the followed Detailed Description of the Invention which, taken together with the appended Figures, illustrate an exemplary embodiment of the invention and serve to explain the principles of the invention.

THE FIGURES

Figure 1A shows in block diagram form the operation of a RAIS daemon within a host handling various object types in accordance with the present invention.

Figure 1 B shows the linking of various service monitors to their associated test objects and load objects.

Figure 2A shows in block diagram form a network comprising a plurality of service groups.

Figure 2B shows the data structure of a Control Info Message.

Figure 2C shows the data structure of the Host Information Entry.

Figure 2D shows the data structure of the Service Address Database.

Figure 3 shows in flow diagram form the "transmit" thread of the service monitor.

Figure 4A shows in flow diagram form the "receive, thread of the service monitor.

Figure 4B shows in flow diagram form the "acquire message portion of the receive thread.

Figure 4C shows in flow diagram form the Receive message portion of the receive thread.

Figure 4D shows in flow diagram form the "update tables" portion of the receive thread.

Figure 5A shows in flow diagram form the "scripts" thread of the service monitor.

Figure 5B shows in flow diagram form the Acquire action, portion of the scripts thread.

Figure 5C shows in flow diagram form the "release action" portion of the scripts thread.

Figure 6A shows in flow diagram form the "control; thread of the service monitor.

Figure 6B shows the data structure of the control service address table.

Figure 6C shows the data structure of a table showing utilization and weighted utilization.

DETAILED DESCRIPTION OF THE INVENTION

The present invention provides a system and method by which a computing service may be provided by any of a plurality of systems within a group, such that the computing service provided by that group will continue to be available as long as at least one system is available within the group. The invention is particularly well-suited to operation on a Redundant Array of Independent Servers, or RAIS.

In addition, the present invention permits new hosts to be added and configured without reconfiguring existing hosts. The system of the present invention then dynamically and automatically incorporates the new hosts into the appropriate group. Similarly, failed hosts may removed without reconfiguring the remaining hosts.

The foregoing improvements are achieved through appropriate management of the IP addresses of a plurality of computer systems, each of which may be thought of as a service provider for a computing service available from the network. Each computer system, or host, is assigned a unique IP address in the conventional manner, but the IP addresses are then grouped according to providers of a desired service, such that each member of the group is available to provide that service. This collection of hosts may be referred to a "service group, and the members of a service group cooperate to provide a particular service. A host may belong to multiple service groups, but for purposes of clarity and simplicity the following discussion will be limited to describing the interaction among hosts in a single service group.

To maintain coordination among the various members of the service group, a "service group" address, is established, which is simply an IP address for that group. All members of the service group communicate with each other using the

service group address -- that is, ach member transmits messages to all other group members by addressing the message to the service group address, and each member receives messages from all other service group members by receiving messages destined to the service group address. These messages, which may be thought of as "control messages," communicate the well-being of the host providing the message, the status of each service address, actions to be performed by the members of the service group, and any other function suited to the messaging capability.

For increased reliability, multiply redundant hardware network interfaces may be provided for use by the hosts. These interfaces, which- may be referred to hereinafter as 6control interfaces," provide the means by which the host may send and receive control messages. Certain network interfaces may be referred to as 4service network interfaces," which permit clients to communicate with the service on the host. Thus, service network interfaces are simply network interfaces used by clients to access the service on the host. A network interface may be both a service interface and a control interface; however, in a presently preferred arrangement each host will typically have at least two control interfaces, each connected to an independent and isolated network.

A Control address," as used herein, refers to the IP address of a control interface. Each host will preferably have one or more control addresses, and typically must have at least one control interface. The first control address is typically used to identify uniquely the associated host, and may also be referred to as the "host id" or "id_address" hereinafter. Control addresses identify which interface the host should use to transmit and receive group messages.

A service address is an IP address that hosts in the service group provide service through. Management of such service addresses is a key portion of the present invention.

When a new host becomes available to a service group, it introduces into the service group both a host id and a service address. The service address is associated with, or belongs to, this host unless the host fails, and is known as the Preferred service address for that host.

As noted previously, each host within a service group periodically sends control (or info) messages to the service group via the service group address. Each info message includes, among other things, a set of the hosts control addresses, its preferred service address and a set of service addresses it is currently serving. These messages enable members of the service group to determine the status of each host in the service group.

Another function of the control messages is to permit the hosts within a service group to elect one of the hosts to fill the logical role of "leader" of the service group. Because the role of leader is logical, the role may move from one host to another at various times, including when hosts are added or removed. In an exemplary embodiment, the host with the lowest host id is denominated the leader, although other methods are also acceptable.

The leader of a service group manages service addresses within the group. Among other tasks, the leader tries to establish various conditions at equilibrium, including that all service addresses are being served, that each service address s assigned to only one host, and that each host will serve its preferred service address. The leader also initiates corrective action if the conditions are violated, typically by causing a host either to serve or stop serving a service address; this may be thought of as "acquiring" or "releasing" By such acquiring and releasing Of service addresses, the hosts included within a service group ensure that all addresses are being served. In addition, the present invention includes the ability to modify domain name service bindings by modifying zone files and notifying the DNS server of zone file changes, as described in the concurrently filed U.S. Patent Application identified as number (1) in the Related Applications portion, above. Domain Name Zones are also discussed at length on the Internet RFC pages 1034 and 1035. Basically, a zone is a group of IP addresses with a single common parent, although a zone may not include all of the child addresses or any lower level nodes.

The leader is also solely responsible for assigning service addresses; thus, while a host added to the system presents with a preferred service address, it does not own that service address until the leader assigns it. The leader listens

to control messages and quickly releases the preferred service address introduced by the joining host if currently being served by another host by another host in the service group. If the preferred service address is not being served by anyone, the leader simply assigns it to the new host.

The case where a different hoot has already been assigned the preferred service address of the new host usually arises where a host has previously failed, and is now rejoining the group. In such a situation, the leader will typically request the host currently owning the preferred service address to release it, and will then wait for the release request to complete before assigning the preferred service address to the joining host.

When a host fails, the leader will detect that the service addresses that were originally served by the failed host are not being actively served, because the control messages will show increasingly long periods since the last time the service addresses were served. The leader then reassigns these unserved addresses to other available hosts within the group. The reassignments may be made randomly, or may be made base on a load-balancing scheme, or through any other suitable method.

The reassignment of service addresses permits failed hosts to be removed from the service group. A service address that is not assigned to its preferred host is referred to as an "orphaned" service address. Associated with each orphaned service address is a logical countdown timer maintained by the leader. This countdown timer is created and started when the address becomes orphaned; when H expires, the service address is invalidated. When the service address goes invalid, the leader notices that a host is serving an unknown service address and requests that host to release it.

Because the integrated name service (discussed below) does not advertise orphaned service addresses, such orphaned addresses cannot acquire new clients through the name service. As the clients of the orphaned address logout or disconnect over time, the need from the orphaned service address is eventually eliminated and the address can be invalidated or eliminated. If it is

preferred not to cause such automatic removal of such addresses, the countdown timer can be set for very high values, perhaps in the range of years.

The integrated name service mentioned above refers to the ability of the present invention to modify DNS bindings by modifying zone files and notifying the DNS server of the zone file changes. See the applications described in the Related Applications portion, above; - in particular, see U.S. Pat. ppn. S.N. 08/673,951, entitled "A Service For a Redundant Array of Internet Servers" of Swee Boon Lim, filed 7/1/96.

To ensure that a host hosting a DNS server learns about the members of a service group, H must also belong to that service group. If it is also providing the service of that group, H will receive messages in the normal fashion. If not, it may join the group but simply identify itself as failed. In this manner it will receive appropriate messages, but will not be assigned any service addresses by the leader. In a preferred embodiment of the invention, each service group has its own DNS zone. This requirement isolates the DNS changes originating from a service group to a single DNS zone file. In addition, the DNS zone is set up such that only preferred service addresses of available members are advertised. Since orphaned service addresses cannot be obtained, orphaned service addresses will fade away as described above.

Load balancing is another important aspect of the present invention. Two techniques may be used in combination: either statistical load-balancing, or zone modification. To use zone modification load balancing, the info message includes two values: one is the current load; the other is current capacity. Current capacity is typically expressed in instructions per second; current load is typically expressed in number of instructions per second not idle, measured over a suitable duration. For a network bandwidth intensive service, capacity could be the maximum bandwidth of the service interfaces, while load may be the average bandwidth consumed during the last measurement interval.

Utilization is simply determined by dividing the load by the capacity. An operator-configured high water mark can be established, and if a given host exceeds the high water mark for utilization, that host can be removed from the

DNS zone in most circumstances. This shifts new connections to other hosts with less utilization. When the hosts utilization drops below an operator-configured low water mark, the host is added back to the DNS zone and again becomes available for assignment of new addresses.

A configuration parameter may also be provided which specifies the minimum number of available service addresses that must be present in the DNS zone, to prevent removal of all preferred service addresses from the zone. To prevent the number of available addresses from falling below this minimum, highly utilized hosts may be advertised.

By proper management of the service addresses, various network functions can be managed, hot sparing, quiescing and failure detection. Hot sparing involves hosts designated as service providers only if another host fails; in essence, such devices are surrogates for failed hosts and their preferred service addresses. A hot spare host does not have a preferred service address, but instead sends info messages advising that it is available. This allows the leader to assign service addresses to the hot spare in the event it is needed. In particular, the leader will typically assign orphaned service addresses to hot spares only. Assignment of service addresses may either be random or in accordance with a load-balancing scheme.

Quiescing is defined as preventing a host in a service group from acquiring service addresses, while at the same time permitting that host to release addresses; in an exemplary embodiment, setting a host as quiesced requires intervention by the system operator. This enables system administrators to address resource problems without declaring the affected host as failed. Like failed hosts, quiesced hosts are not included in the service groups DNS zone, and thus their address does not show up in the list of available hosts; unlike failed hosts a quiesced host can continue to serve its current service addresses.

Failure detection, also available to be performed by the present invention, involves monitoring of the "info" or control messages sent by each host. The info messages basically serve as a "heartbeat" signal. A total host failure is detected if no heartbeat signal occurs within a timeout interval. The frequency of

transmission of the info message/heartbeat and the timeout interval may be adjusted by the operator to optimize system operation. Additionally, a host may experience service failures, which occur when a host can no longer provide a service due to partial failure. These failures may be transient or persistent, and may involve software or hardware failures such as resource exhaustion or corrupted data, among others. In an exemplary embodiment, hosts with service failures may continue to participate in a service group, although such hosts will voluntarily release their service addresses.

Referring first to Figure 1A, a computer system, or host, 10 has resident therein a daemon 20 handling network services 30A and 30B such as HTTP or NEWS. Each of such services 30 comprises a plurality of object types, including a service monitor SM1 or SM2, a test object TO1 and TO2 (respectively) and a load object LO1 and LO2 (respectively.) As will be appreciated in greater detail hereinafter, the Service Monitor includes many of the key program elements in the present invention, and essentially manages many of the remaining operations. The test object serves simply to verify that the particular host is working correctly and makes this information available for use by the service monitor. Likewise, the load object simply measures capacity and load so that this data can be used by the service monitor. The combination of object types cooperate to provide the network service; the service monitors, test objects and load objects are implemented by processors 32 executing in memory 34.

Although only two such services are shown in Figure 1A, it will be understood by those skilled in the art that any number of services may be provided by a particular host 10.

Referring next to Figure 1B, the relationships between the service monitors SM1 and SM2 and the associated test objects TO1-TO2 and load objects LO1 and LO2 are illustrated in the context of the RAIS daemon 20. In particular, as an exemplary arrangement only, it can be seen that service monitor SM1 is linked to test objects TO1 and TO2 and load object LOT, while service monitor SM2 is linked to test object TO3 and load object LO2. A service monitor may be linked to

any number of test objects, and each test object and load object may be linked to more than one service monitor.

In accordance with the present invention, each provider of a service 30 is identified by a unique IP address. Next referring to Figure 2A, and in accordance with a key feature of the present invention, each network service 30 may be provided by a plurality of hosts 10 of the type shown in Figures 1A-1B. This plurality of hosts is configured in what is referred to herein as a service group 100, and multiple service groups 100 may be interconnected within a network 110 to provide all necessary computer services. A single host 10 may belong to multiple service groups 100; however, for purposes of clarity, the present description will describe the interaction among hosts in a single service group.

To maintain coordination among the various members of the service group, a new IP address referred to as a service group address 105 is established for each service group, and is typically an IP multicast address and a port number. All hosts 10, or members, within a service group 100 communicate with one another using the service group address 105 that is each member transmits messages to all other group members by addressing the message to the service group address, and each member receives messages from all other service group members by receiving messages destined to the service group address. These messages, which may be thought of as control messages, are typically broadcast on a periodic basis by each host using a protocol such as UDP and communicate a variety of data. The data structure of the control/info message is shown in Figure 2B. Typical message contents include: (1) the well-being of the host providing the message; (2) capacity and load; (3) a list of control addresses; (4) the preferred service address; (5) a list of addresses currently being served, including (a) the service address, and (b) the state of that service address, which can be either acquiring, releasing, on-line, or not served; and (6) a list of addresses that should be served including (a) the service address and (b) the last time that address was served by the preferred host.

The foregoing data about the transmitting host which is included in the control message is typically maintained in a data table referred to as a Host

Information Entry, and is transmitted only if authoritatively correct. An exemplary Host Information Entry structure is shown in Figure 2C. In addition, in an exemplary embodiment the control message further includes whatever information that host has received about the other hosts included in the service group, and when information was last received from that other host (i.e., aging.) This data is stored is a database, referred to as the Host Information Database. The data structure of Host Information Database is simply a series of Host Information Entries, organized by id_address. In addition, the Control Message includes a Service Address Database, which is arranged as a set of service addresses and the time since last update for each; See Figure 2D for an exemplary structure. As will be appreciated hereinafter, the time is reset with every new entry; and such resets are used to ensure that the host responsible for that service address has not died. A failed, or dead, host stops sending messages, which alerts the present invention to reassign that hosts tasks.

The transmission of the control messages is handled by the Service Monitor associated with each service, such as SM1. The Service Monitor is multi-threaded, for which the threads are created for: Transmit, Receive, Script, Control and DNS. The Transmit thread transmits the control message, while the Receive thread receives control messages from other hosts within the service group. The Script thread handles all events that may take too long for handling by the other threads, while the control thread determines which of the hosts within the service group will be "leader" and will make assignment decisions. Finally, the DNS thread simply updates the DNS. Each of these threads will be discussed in greater detail hereinafter.

Referring next to Figure 3, the Transmit thread may be better understood. It will be appreciated that each host begins one Transmit thread for each service. The Transmit thread starts at step 300 in the "Sleep" state. The process advances at step 305 by invoking the test objects TO1...TOn associated with that service, followed at step 310 by checking whether the test is passed. If the tests fail, or a NO results at step 310, the process advances to step 320 and invokes a script to schedule the release of all current service addresses (so that the service

addresses can be reassigned to another, non-failed host), after which a control message is created and sent at step 330. The process then loops back to sleep at step 300.

However, if the tests at step 310 resulted in an OK, the process advances to step 350 where load and capacity information (discussed above) is gathered from each of the load objects LO1...LOn. At step 360, the service address and timer data is then written if an Im alive time-out timer requires it. A control or info message is then created and sent at step 370 to communicate the state of the host to the remaining members of the service group, and at step 380 ARP packets are sent. The process then loops back to sleep at step 300. Thereafter, the daemon will restart the process after an appropriate time.

Reference is now made to Figures 4A4D. The Receive thread is shown at a top level in Figure 4A, with Figures 4B-4D showing the various branches that are included as part of the Receive thread. The Receive thread initiates at step 400, and advances to step 402 where a message is received from another host within the service group. A check is then made at step 404 to determine whether the message is a duplicate; if so, the process loops back to step 402 to receive another message. If the message is not a duplicate, the process advances to step 406 and checks to see whether the message is an acquire message. If so, the process branches to Figure 4B, discussed below. If not, the process advances to step 408 and checks to see whether the message is a release message. If so, the process branches to Figure 4C, discussed below; if not, the process advances to step 410. At step 410 a check is made to determine whether the message is an info, or control, message. If so, the process branches to Figure 4D, also discussed below. If not, the message is discarded and the process loops back to step 402.

Referring next to Figure 4B, the operation of the Receive thread in response to an "Acquire, message can be seen. The Acquire" process basically involves having the leader send a message identifying the source and destination id_addresses, together with which service address is to be acquired. Thus, the process starts at step 412 by checking whether the received acquire, message is

for that host. If not, the process is done and returns at step 414 by looping to step 402. If the message is for that host, the process advances to check whether the message is from that hosts leader by comparing addresses in the Host Information Database. Again, if not, the process again returns at step 414.

If the check at step 416 yields a YES, the process advances at step 418 and determines whether the receiving host is OK. If not, the process again returns at step 414; but if so, the process advances to step 420 and a check is made to determine whether the host is quiesced (which can be ascertained from the Host Information Entry, although operator intervention is typically required to set the host as quiesced.) If the answer at step 420 is yes, no service address can be acquired, so the process again returns at step 414. However, if the host is not quiesced, a check is then made at step 422 to determine whether the host currently is servicing the service address identified in the acquire message. This can be determined by examining the hosts Host Info Entry. If so, there is no need to reacquire the "new" service address, and the process returns at 414. If the host is not currently servicing the identified service address, the process advances at step 424 by updating the Host Information database with an entry for the new service address, and an acquire action is scheduled at step 426. The "acquire" action is described in greater detail in connection with Figure 5B. The process then returns at step 414.

Referring next to Figure 4C, the Release portion of the Receive thread can be better appreciated. From the following, it will be apparent to those skilled in the art that the Release portion is substantially similar to the Acquire portion. The process starts at step 432 by checking whether the message is for the receiving host. If not, the process returns at step 434. If so, the process advances to step 436 and checks whether the message is from the leader of that service group by comparing Host IDs in the Host Information database; if not, the process again returns at step 434. If the message is from the leader, then a check is made at step 440 (by looking at the relevant flag in the Host Info Entry) to determine if the identified service address to be released is on-line. As with the "acquire" message, the "Release message comprises source and destination

id_addresses. If the service address to be released is not being serviced by the receiving host, the process returns at step 434. However, if the service address is being serviced, as determined by the listing in the Host Info Entry, the service address is first marked as "releasing" in the Service Address Information, and the Host Information database is updated at step 442 by deleting that service address from the list of service addresses serviced by that host, and the process advances to step 444 to schedule a release action. A release action is described in connection with Figure 5C. The process then returns at step 434.

Referring next to Figure 4D, the response to an info, or control message can be understood. The response to an info message is simply to update the appropriate tables maintained by the receiving host, particularly the Host Information database referred to previously, which effectively updates the status of each service address. Thus, the process begins at step 450 by updating the Service Address database with the information received from the other host, and then advances to step 452 by updating table, or list, of addresses that should be served by the service group. The list specifies, for each service address, the last time that address was served by the preferred host, and resets a time-out timer that basically serves to confirm that the responsible host is alive and well. Once these tables are updated, the process returns at step 454.

The next thread of the Service Monitor to be considered is Scripts, for which reference is made to Figures 5A-5C. Figure 5A shows the overall thread, while Figures 5B and SC show key branches of the thread. Scripts are generally more complicated processes, and because of this may be run after some delay rather than immediately. The process begins at 500 and advances to step 502 by removing the request for action from the queue. A check is then made at step 504 to determine whether the requested action is an "Acquire" action. If so, the process branches to Figure 5B. If not, the process advances to step 506 to determine whether the requested action is a "Release" action. If so, the process branches to Figure 5C; if not, the process advances to step 508, where a check is made to determine whether the action requested is a test failed action. If the check results in a YES, a "Test Failed" SNMP trap is sent at step 510. If the

check results in a NO, the process advances to step 512 where a check is made to determine whether the requested action is a "Cannot Assign" action. If the check results in a YES, the process branches to step 514 and a "Cannot Assign" SNMP trap is sent. If the answer is a no, the process completes at step 516.

Turning next to Figure 5B, the "Acquire" branch from Figure 5A may be better understood. As noted previously, the acquire process assigns a service address to a host. The branch begins at step 530 by configuring the hardware service interface for the software service address, and then advances to step 532 where the Acquire script is run. Following execution of the Acquire script, the process advances to step 534 and a check for errors is made. If no errors occurred, the process advances to step 536 and updates, for that service address, the Host Info Entry to On-line. The process then advances to step 538 and sends an SNMP Acquire trap, following which the process completes at step 540.

If an error occurred at step 534, such that a YES was returned, the process advances to step 542 by configuring the service interface hardware to remove the service address. The process then continues at step 544 by updating the Host Info Entry for that service address to "not serving" and then, at step 546, sends the "Acquire fail" SNMP trap. The process then completes.

Referring next to Figure 5C, the Release script branch of the Scripts thread can be better appreciated. Similar to the Acquire branch shown in Figure 5B, a release action causes a host to release one of the service addresses it has been handling. The branch begins at step 560, and runs the Release script. The service interface (hardware! is then configured at step 562 for the service address (software) to remove the service address being released. A check for errors is made at step 564, and if an error occurs a Release Fail trap is sent at step 566, after which the process completes at step 568. Whether or not an error occurs, a "Release" trap is sent at step 570 the Host Info Entry is also updated to remove the affected Service Address and the process completes.

With reference to Figure 6A, the Control thread of the service monitor can be better understood. The purpose of the Control thread is to establish which

host within the service- group will be the leader, and to permit the leader to manage service addresses dynamically in accordance with the state of the system. The process starts at step 600 in the Sleep mode. When awakened, the processes advances at step 602 to determine who the leader is currently A check is made at step 604 to determine whether the current host is the leader If not, the process loops back to step 600.

However, if the check at step 604 yields a yes, the process advances to step 606 and removes old, timed-out entries and silent host information by updating the database to the current state. A silent host is a host that has failed to send a control info message within the time-out period, such that the leader assumes the host to have died. This permits additional decisions, such as reassignment of the service addresses formerly served by a failed host to be made reliably by the leader. Once the outdated entries are removed, a control database is built at step 608, discussed in greater detail below. After the control database has been built, each of the service addresses is processed at step 610 to update its status to current information, also described in greater detail below. The process then loops back to the Sleep mode at step 600.

The control database basically comprises three tables. The first is the control service address table, shown in Figure 6B, which sets forth four items: (1) the service address within the service group, (2) the states of each service address (i.e., acquiring, releasing, on-line, or unserved), (3) the preferred service address (or preferred id_address) of each host, and (4) a set of id_addresses which claim to service that service address. The second table is the control Spare_Host_List and the third is the Non_Spare_Eligible_List; these are simply a single-column list of IP addresses. For hot spares, the preferred service address is operator-configured to zero, or not assigned.

The table is built through a pair of nested loops, beginning with an examination of the Host Information database. A pseduo-code representation for the outer nested loop of an exemplary embodiment is:

if host is OK

```
if preferred service address = 0
        if not quiesced
                Add Host to Spare_Host_List
else
                if not quiesced
        Add Host to Non_Spare_Eligible_List
endif
if preferred service address is not expired,
        set preferred id_address of entry for
preferred_service_address
endif
```

The inner nested loop is, for each service address in the list of service addresses currently being served, as maintained in the Host Information Entry currently being processed:

```
if service_address not expired
        add id_address to id_address_list for service address
        add state of service address
else
        send release message to host to release expired service addresses
endif
```

As used in the foregoing, "id_address" means the Host ID. As will be appreciated by those skilled in the art, the foregoing iterates through the preferred service addresses and id_addresses, or items (3) and (4) of the control service address table, and then returns to iterate through the states of the various service addresses in the table.

A pseudo-code representation of the steps needed to develop the states of the various entries is as follows:

```
if state including acquiring or releasing,
        continue
```

```
endif
if number of id_addresses [item (4) in Control Service Address table] = 0
        then unserved
        continue
endif
if has preferred id_address and service address not served by
        preferred id_address and preferred id_address has not quiesced
then
        add preferred id_address to entry
endif .
if number of id_addresses > 1,
        then duplicate exists
endif.
```

The foregoing process essentially injects an entry of a service-address artificially, and relies on a later check for duplicate hosts to remove one of the two hosts now serving that address. "Duplicate" as used in the pseudocode is a function and requires involvement of the system to resolve duplicates, as will be discussed in greater detail hereinafter.

Pseudocode for managing unserved states can be represented as follows

```
if preferred_id_address != 0 and [where !=, means not l]
        preferred host is not quiesced then
        choice = preferred_id_address
else if (spare_host count > 0) [from Spare_Host_List Table]
        choice = random pick from Spare_Host_List
else if (Non_Spare_Eligible count  O)
        choice = random pick from Non_Spare_Eligible_List
else schedule cannot assign action
        done/return
endif
send acquire message to choice
```

```
done
```

To ensure proper operation, a- check must be made by the leader for duplicate hosts serving the same service address, and one of the duplicates removed. As previously noted, in an exemplary embodiment all control decisions are made by the leader. A pseudocode representation for such a process is:

```
if preferred_id_address != 0 and preferred host not quiesced
        choice = preferred_id_address
else
        if service address served by one or more from Spare_Host_List
        choice = random pick from Spare_Host_List serving
Service_Address
        else
        choice = random pick from id_addresses
endif
sendmsg to all others (except choice) id_addresses to release.
```

The foregoing process can be seen to assign the service address to the preferred service address if that address is available; otherwise, the leader assigns the service address to an available hot spare. If neither of these works, the leader can simply pick one. The leader then tells the remaining service addresses to release the address. An exception to this process exists for the artificially injected address discussed above; in this instance, the artificially inserted address is not told to release.

The final thread of the Service Monitor is the DNS thread. The DNS thread basically determines load versus capacity for load balancing, and is calculated from a comparison the utilization of the various service addresses both currently and historically. A first table or database M, shown in Figure 6C, stores the historical utilization and weighted utilization for each service address, while a second table or database BB stores current utilization and weighted utilization for the various service addresses. It will be appreciated that databases M and BB

have the same structure, with the data from database BB simply being transferred to database M as a new sample is entered in database BB. The pseudocode representation of the DNS thread can then be represented as follows:

```
Initialize current DB [BB] to empty
For each entry in Host Info DB
    if entry expired [i.e., host timed out]
    or host quiesced
    or preferred_service_address = 0 then
        continue
    endif
    util = (load * scaling factor) / capacity
    add new entry to BB
    if (util  high water mark) then
        weighted_util = util + (delta)
    else if (util  low water mark) then
        if service_address is in M, then
            weighted_util = util
        else
            weighted_util = weighted_util + delta
        endif
    else
            weighted_util = util
    endif
endfor
    if number of entries in BB > minimum number of service addresses
    then sort BB by weighted_util for each entry in ascending order for
    each entry in BB starting at the minimum number of service
    addresses (i.e. index starts at minimum of service address and ends
    at number of entries in BB)
```

```
                if (entry's weighted_util > high water mark) then
                    remove entry
                endif
            endfor
        endif
        if not same list addresses as M then
            update DNS zone
        endif
        swap BB and M [i.e., current becomes last and last will become next
            current
```

It will be appreciated that the foregoing scaling factor can be varied according to optimize performance according to a number of criteria. The scaling factor is typically in the range of 100 to 1000, so that the value of "Util" is an integer. The scaling factor, once selected, will typically remain fixed although some paradigms exist in which variation of the scaling factor may be desirable. The value of Delta is the difference between the high water mark and the low water mark, and is preset.

It will! also be appreciated that the sorting of the entries in the current database BB provides a pruning, for purposes of load balancing.

Having fully described a preferred embodiment of the invention and various alternatives, those skilled in the art will recognize, given the teachings herein, that numerous alternatives and equivalents exist which do not depart from the invention. It is therefore intended that the invention not be limited by the foregoing description, but only by the appended claims.

What is claimed is:

1. A computer program product comprising

a computer usable medium having computer readable code embodied therein for causing reassignment of computer services from one host to any of a plurality of other hosts within a service group, the computer program product comprising

first computer readable program code devices configured to cause a computer to transmit a message setting forth the state of the computer,

second computer readable program code devices configured to cause a computer to receive at least one message from at least one other computer, and

third computer readable program code devices configured to respond to messages to acquire or release computer services.

2. The computer program product of claim 1 wherein the messages to acquire or release computer services are the result of the failure of another host within the service group.

3. The computer program product of claim 1 wherein the messages to acquire or release computer services are the result of load balancing within the service group.

4. A method for dynamically reassigning computing services to various hosts in a network including the steps of

establishing a service group comprising a plurality of host computers, each of which can provide a computing service,

assigning responsibility for a computing service to a first host computer within the service group,

transmitting among the host computers of the service group messages representative of the state of at least the first host computer,

receiving from the remaining host computers within the service group messages representive of the state of such remaining host computers,

evaluating the presence or absence of such messages and their contents, and

reassigning to another host, within the plurality of computers, responsiblity for the computing service in response to such evaluation.

5. The method of claim 4 wherein the reassigning step is the result of a lack of a message from one of the hosts.

6. The method of claim 4 wherein the reassigning step is the result of load balancing within the service group.

7. Apparatus for providing dynamic reallocation of computing services within a network including a plurality of hosts comprising

a service group including a plurality of at least three hosts each available to provide a desired computing service,

a message portion configured to provide data concerning the state of each of the plurality of hosts,

a leader portion configured to assign responsibility for each of the computing services to any host within the service group in response to the message portion.

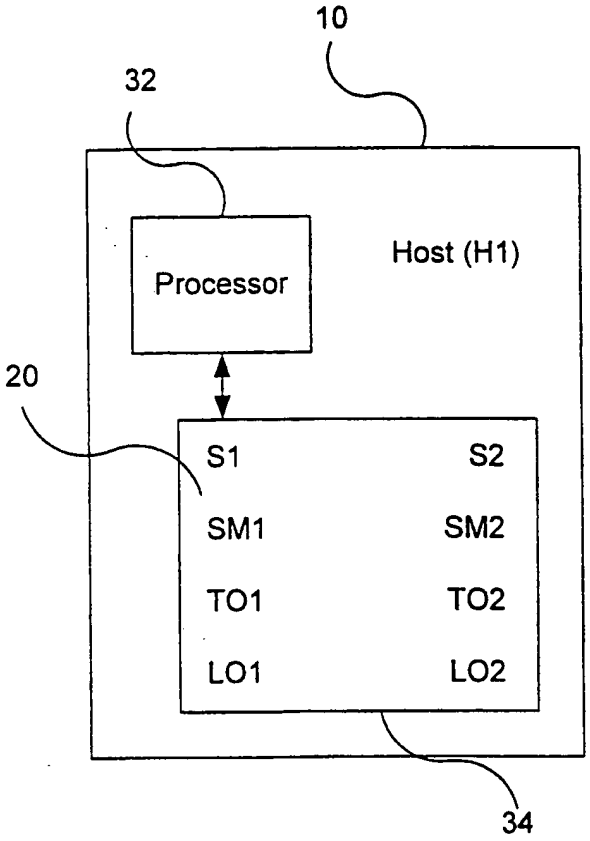


Fig. 1A

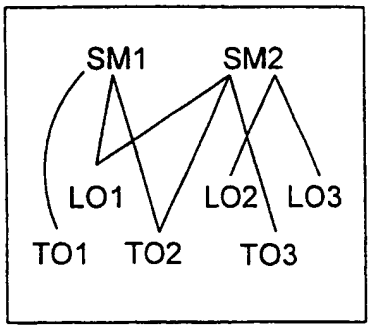


FIG. 1B

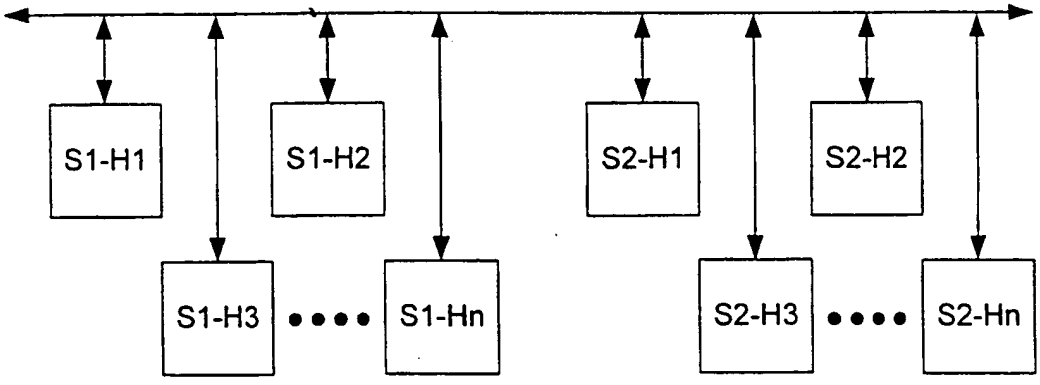


FIG. 2A

MESSAGE SEQ NUMBER	GLOBALLY UNIQUE ASCENDING NUMBER
MESSAGE TYPE = INFORMATION	
MY HOST INFORMATION	HOST INFORMATION ENTRY (A)
OTHER HOST INFORMATION	HOST INFORMATION D B (C) (i.e. SET OF HOST INFORMATION ENTRIES)
SERVICE ADDRESS db	REF OB (SERVICE ADDRESS LAST RECEIVED)

FIG. 2B

ID ADDRESS	(IP ADDRESS)
CONTROL ADDRESSES	(SET OF IP ADDRESSES)
OK	(FLAG) BINARY
QUIESCE	(FLAG) BINARY
LOAD	(UNSIGNED 32-BIT)
CAPACITY	(UNSIGNED 32-BIT)
SERVICE ADDRESSES	(SET OF SERVICE ADDRESS INFORMATION) (INDEX BY SERVICE ADDRESS)

FIG. 2C

SERVICE ADDRESS	(IP ADDRESS)
STATE	ENUM OF RELEASING, ACQUIRING, ONLINE

FIG. 2D

FIG. 3

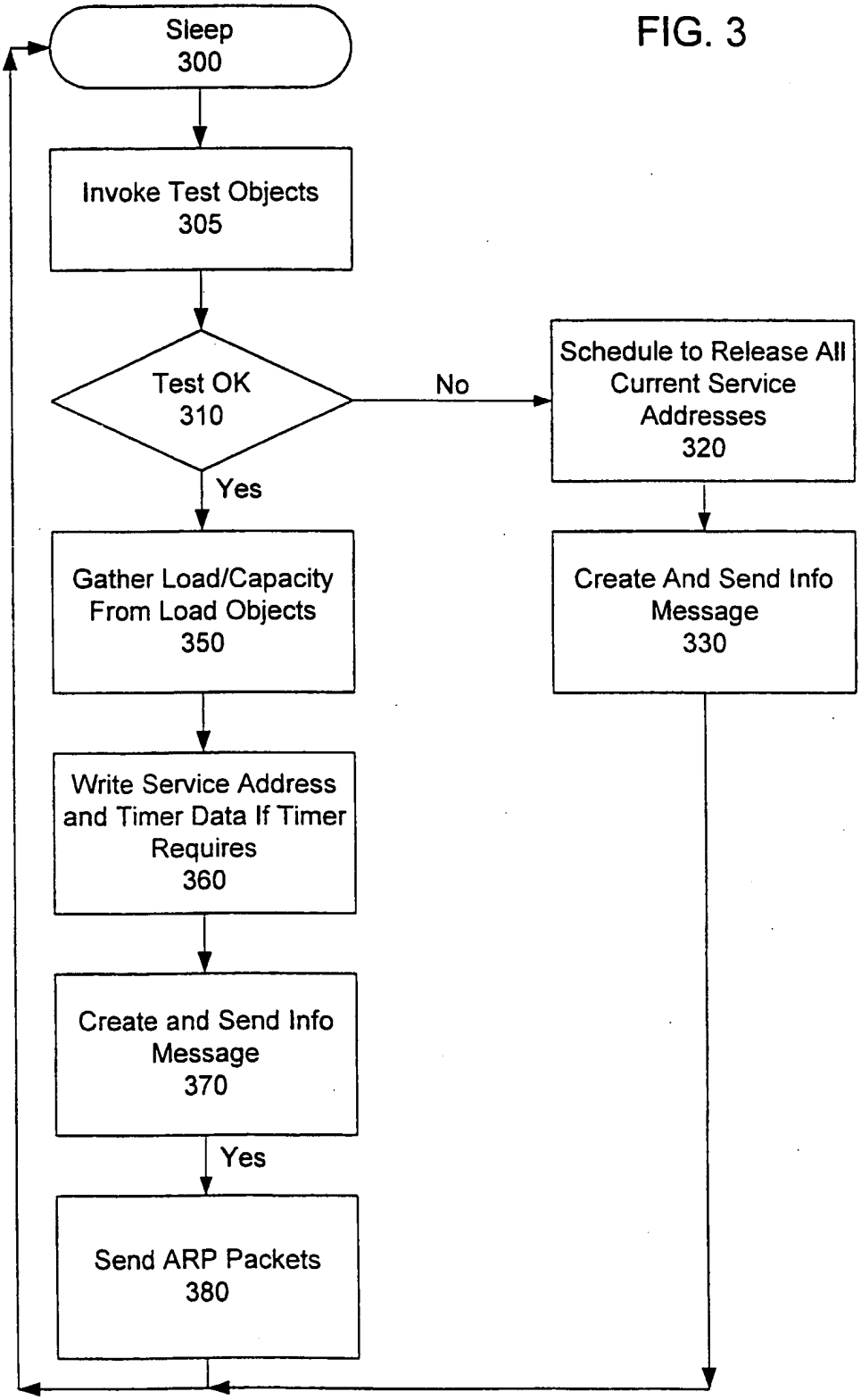
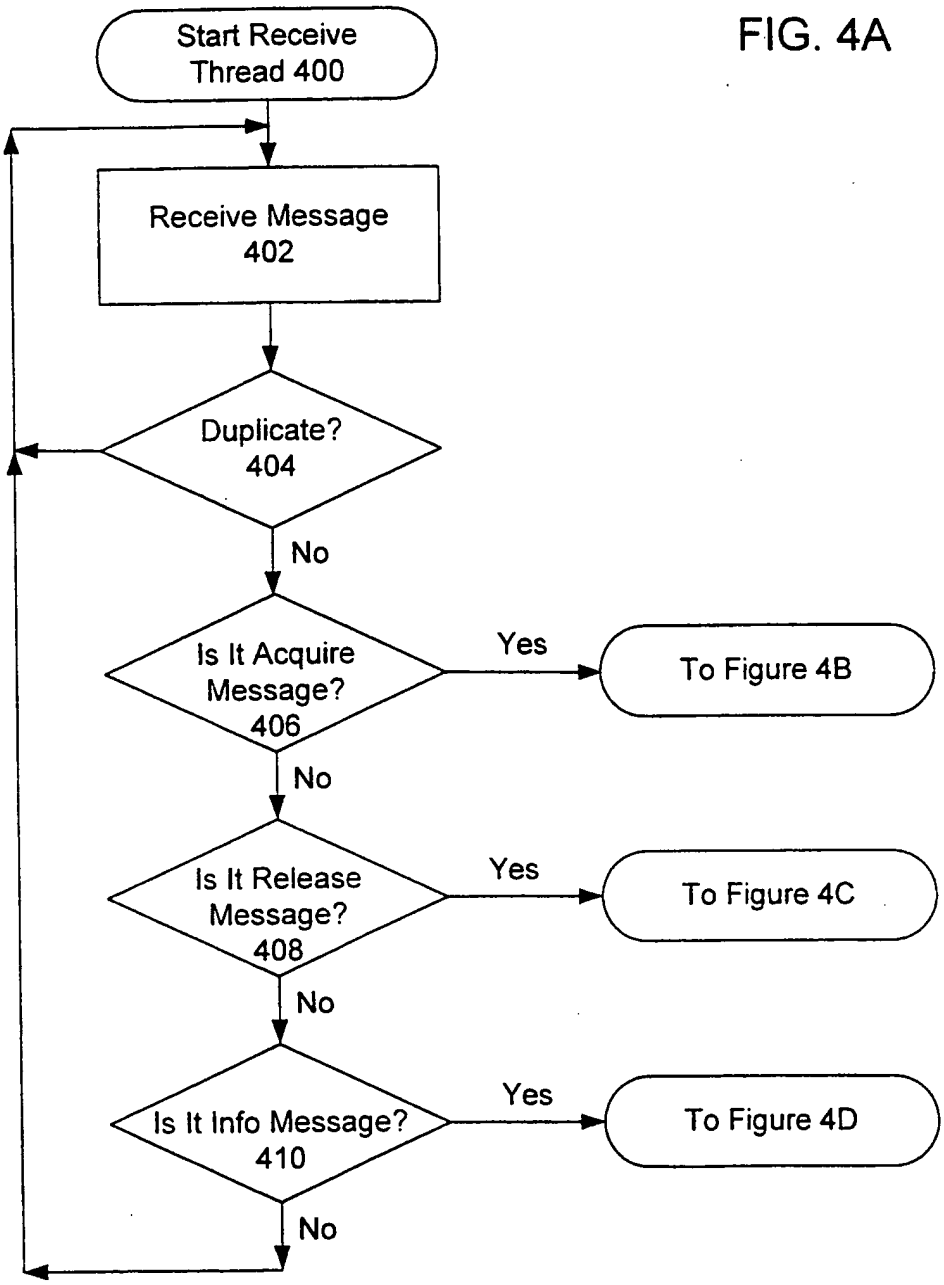


FIG. 4A



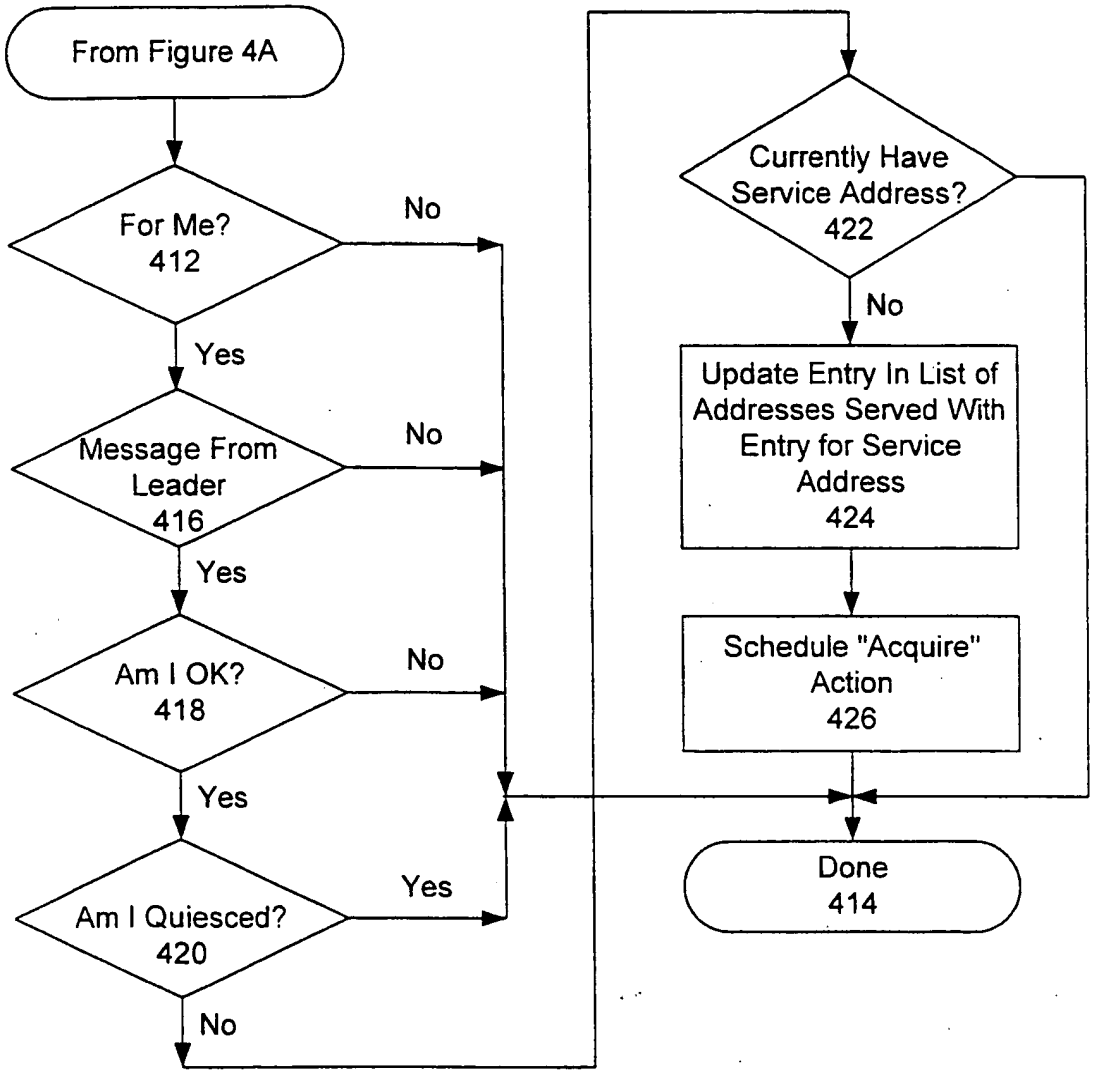


FIG. 4B

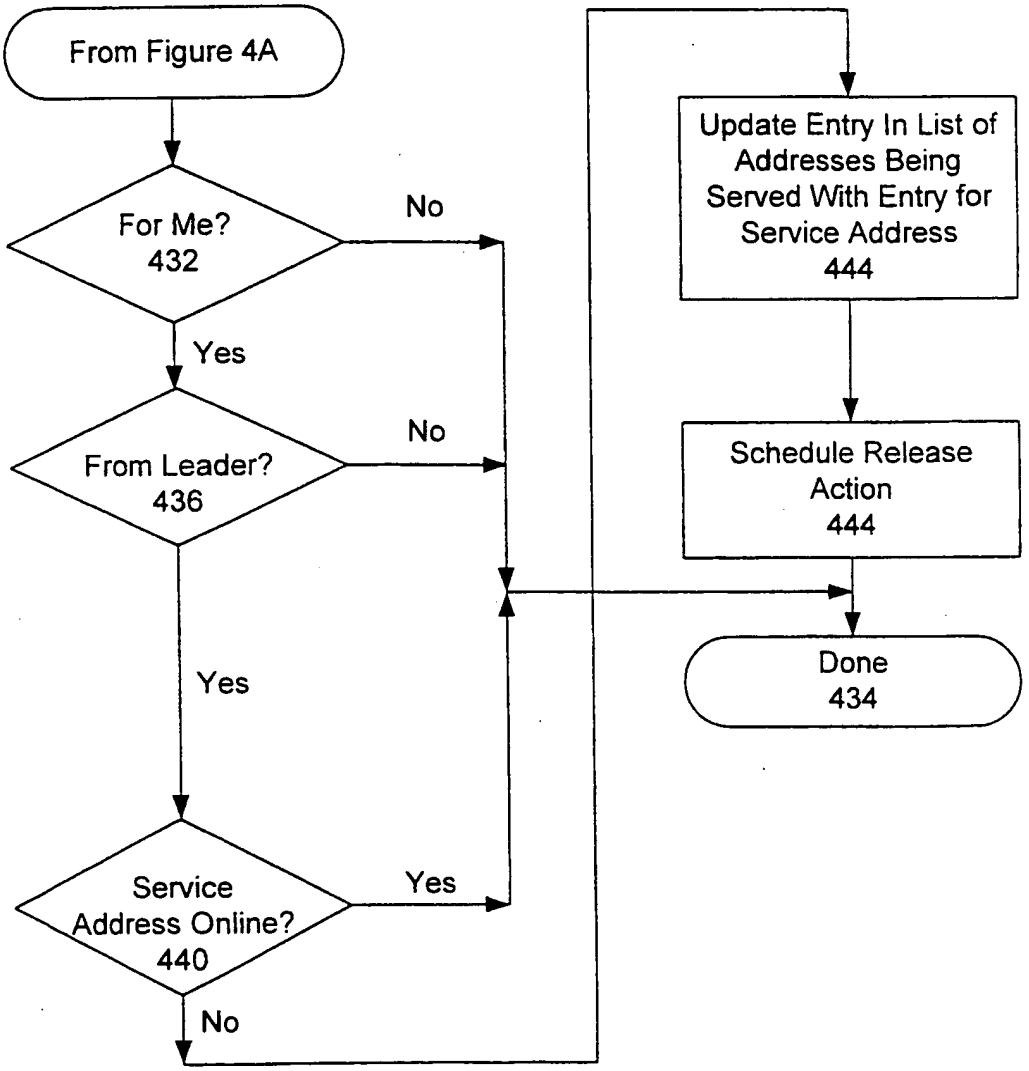


FIG. 4C

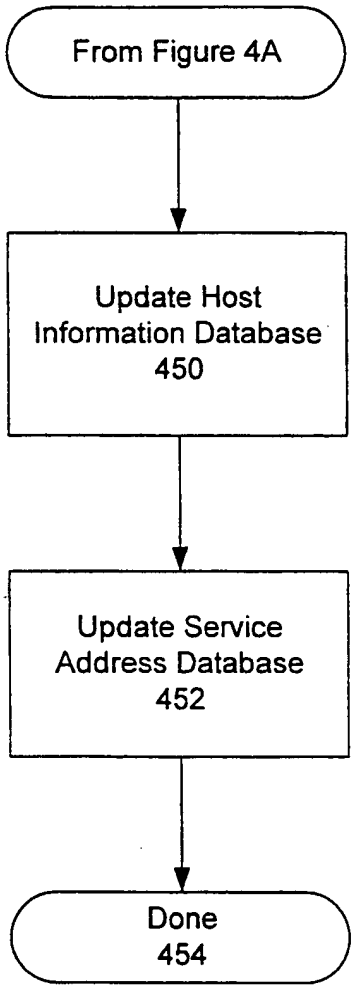


FIG. 4D

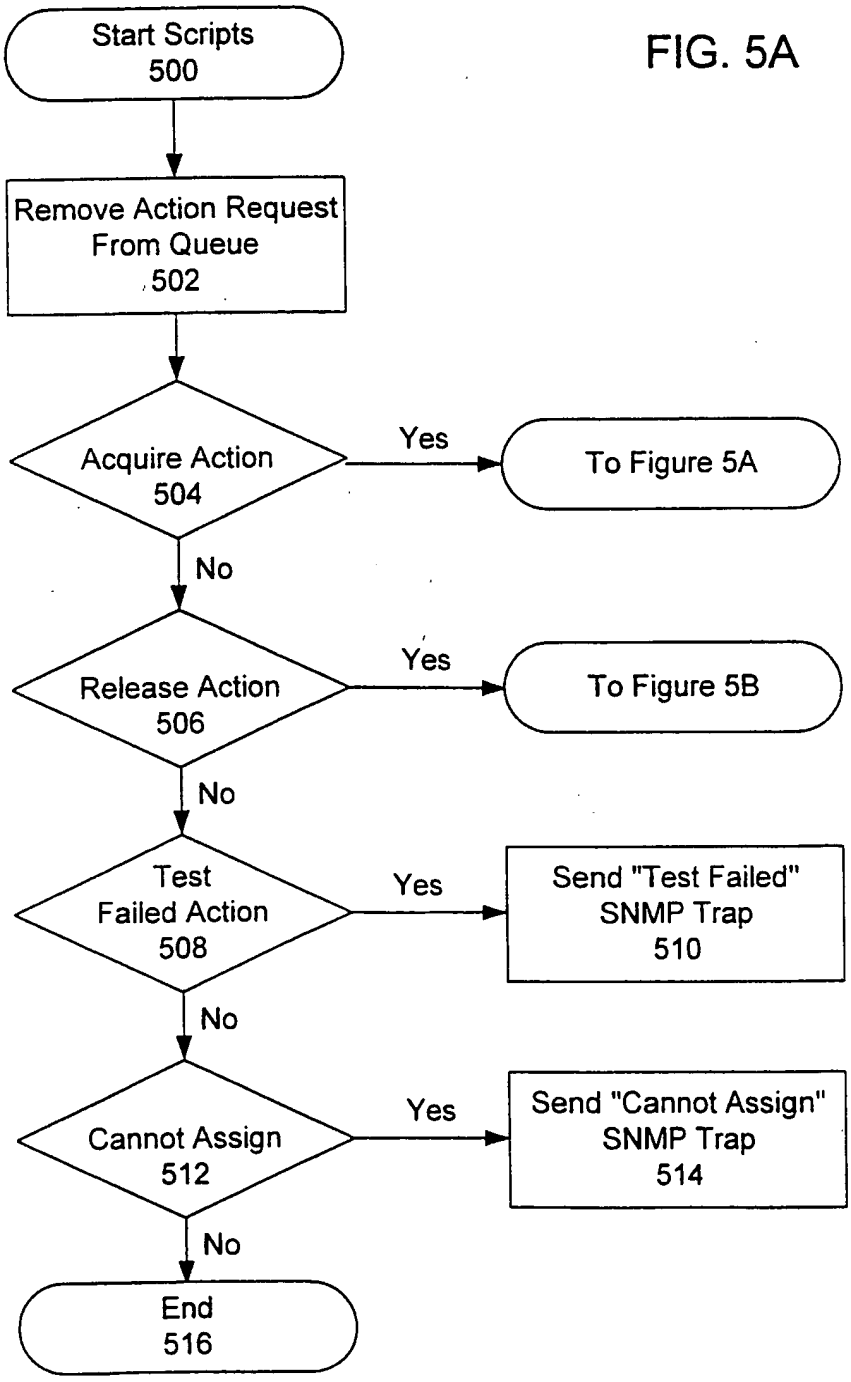


FIG. 5A

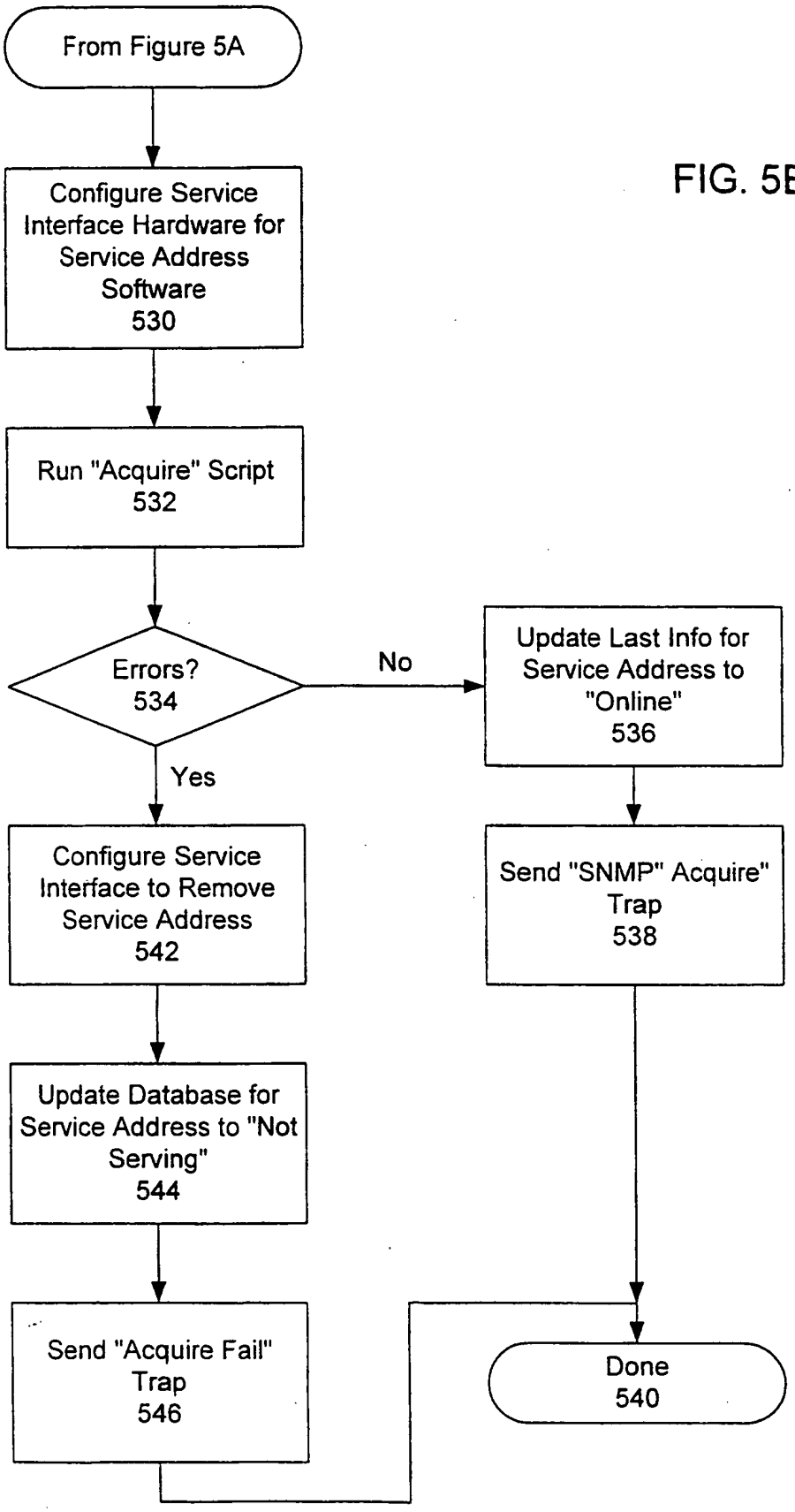


FIG. 5B

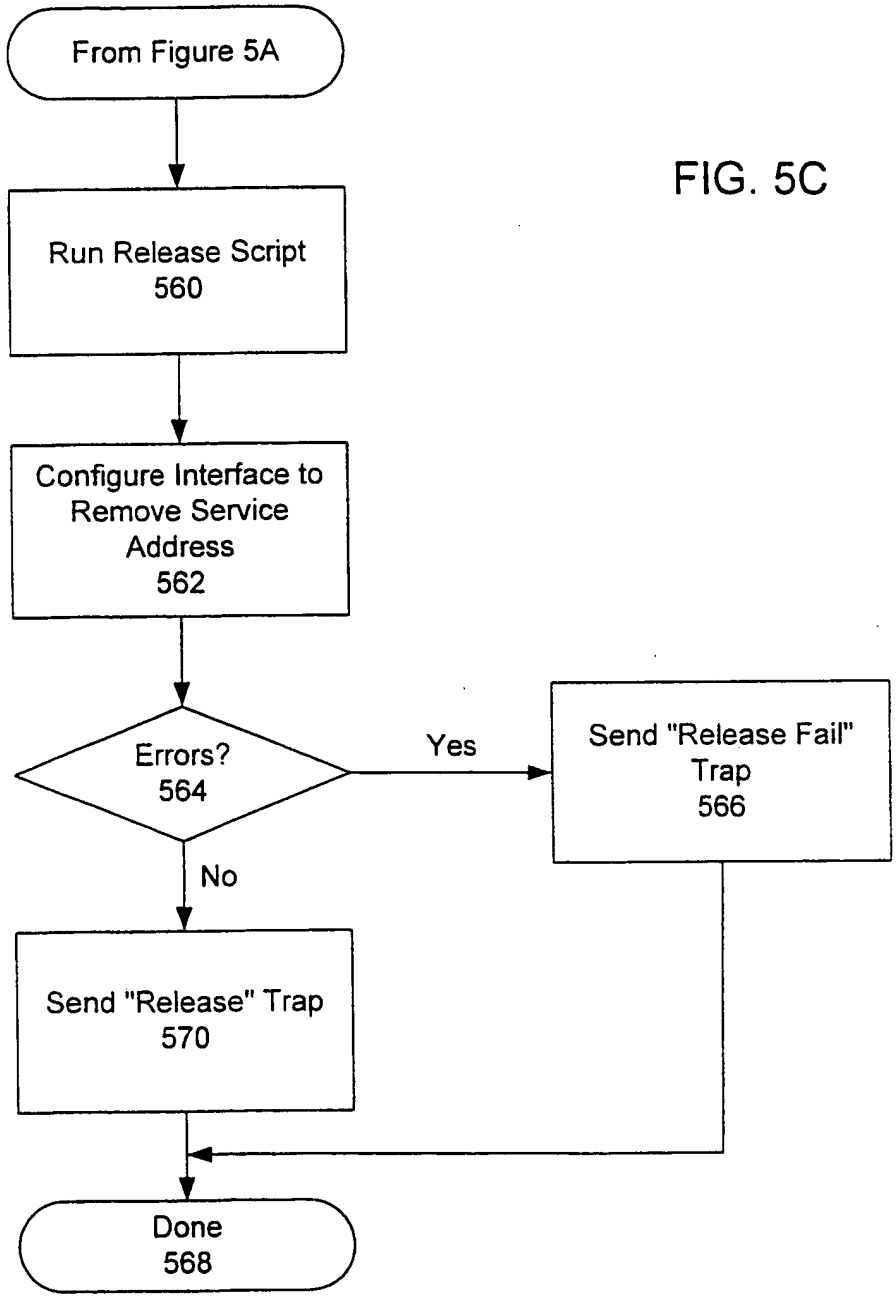


FIG. 5C

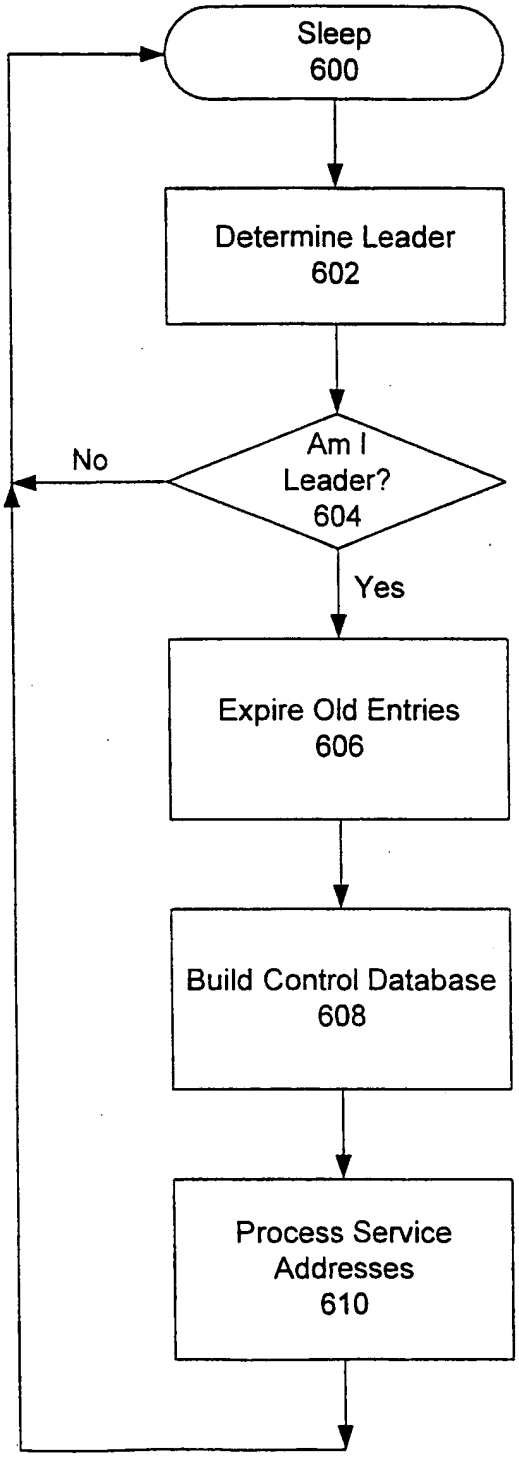


FIG. 6A

SET OF STATES	(BIT MASK ONE EACH FOR ACQUIRING, RELEASING, ONLINE)
PREFERRED-ID-ADDRESS	(IP ADDRESS) ADDRESS OF THE HOST THAT THIS SERVICE ADDRESS IS THE PREFERRED SERVICE ADDRESS
SERVED-BY-PREFERRED	INDICATES IF SERVICE ADDRESS IS SERVED BY PREFERRED HOST / ID ADDRESS (BOOLEAN)
ID-ADDRESSES	SET OF ID ADDRESSES THAT CLAIMS TO SERVE THIS SERVICE ADDRESS (i.e. ACQUIRING, RELEASING, ONLINE STATES)

FIG. 6B

SERVICE ADDRESS	IP ADDRESS
UTIL	UTILIZATION
WEIGHED UTIL	WEIGHTED UTILIZATION

FIG. 6C

INTERNATIONAL SEARCH REPORT

Int .tional Application No
PCT/US 97/22117

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04L29/06 G06F9/46

According to International Patent Classification(IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 H04L G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category '	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
E	EP 0 817 020 A (SUN MICROSYSTEMS INC) 7 January 1998 see abstract see column 8, line 10 - line 27 see column 1, line 22 - line 25 see column 2, line 4 - line 12 see column 2, line 45 - line 50 see column 5, line 44 - line 49 see column 6, line 25 - line 54 --- -/--	1-7

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

' Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

1

Date of the actual completion of the international search	Date of mailing of the international search report
8 May 1998	18/05/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Adkhis, F

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Int tional Application No
PCT/US 97/22117

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JI HUA XIE LI: "A DISTRIBUTED COMPUTING MODEL BASED ON MULTISERVER" OPERATING SYSTEMS REVIEW (SIGOPS), vol. 30, no. 4, October 1996, pages 3-11, XP000639695 see abstract see figure 1B see page 4, line 18 - line 24 see page 5, line 33 - line 36 see page 6, line 4 - line 7 see page 6, line 21 - line 33 ---	1-7
A	ADLER R M: "DISTRIBUTED COORDINATION MODELS FOR CLIENT/SERVER COMPUTING" COMPUTER, vol. 28, no. 4, 1 April 1995, pages 14-22, XP000507856 see page 17, right-hand column - page 22 ---	1-7
A	"LOCAL AREA NETWORK SERVER REPLACEMENT PROCEDURE" IBM TECHNICAL DISCLOSURE BULLETIN, vol. 38, no. 1, 1 January 1995, page 235/236 XP000498750 see the whole document ---	1-7
A	EP 0 384 339 A (DIGITAL EQUIPMENT CORP) 29 August 1990 see abstract see column 2, line 5 - line 11 see column 3, line 31 - line 35 see column 4, line 24 - line 37 see column 14, line 19 - line 26 -----	1-7

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

In ational Application No
PCT/US 97/22117

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0817020 A	07-01-98	NONE	
EP 0384339 A	29-08-90	AT 151183 T	15-04-97
		AU 611605 B	13-06-91
		AU 4996190 A	13-09-90
		AU 630291 B	22-10-92
		AU 7603391 A	15-08-91
		CA 2010762 A	24-08-90
		DE 69030340 D	07-05-97
		DE 69030340 T	20-11-97
		JP 3116262 A	17-05-91
		US 5341477 A	23-08-94